REPORT DOCUMENTATION PAGE

Public Reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comment regarding this burden estimates or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188,) Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE March 29, 2002	3. REPORT TYPE AND DATES COVERED August 2001- February 2002 01 Aug 01 - 30 Jun 03

4. TITLE AND SUBTITLE External Memory Algorithms: Dealing with MASSIVE Data	5. FUNDING NUMBERS 41261-MA

6. AUTHOR(S) Jeffrey S. Vitter	DAAD19-01-1-0725

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Duke University Office of Spponsorted Programs Durham, NC 27708-0491	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSORING / MONITORING AGENCY REPORT NUMBER DAAD19-01-1-0725 41261.3 - MA

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The bottleneck in many applications that process massive amounts of data is the I/O communication between internal memory and external memory. The bottleneck is accentuated as processors get faster and parallel processors are used. Parallel disk arrays are often used to increase the I/O bandwidth. The goal of this proposal is to deepen our understanding of the limits of I/O systems and to construct external memory algorithms that are provably efficient. The three measures of performance are number of I/Os, disk storage space, and CPU time. Even when the data fit entirely in memory, communication can still be the bottleneck, and the related issues of caching become important.

Theoretical work involves development and analysis of provably efficient external memory algorithms and cache-efficient algorithms for a variety of important application areas. We address several batched and on-line problems, involving text databases, prefetching and streaming data from parallel disks, and database selectivity estimation. Our experimental validation uses our TPIE programming environment. Plans for the coming year are to address bottleneck issues in parallel disks, text databases, and XML databases.

14. SUBJECT TERMS Input/Output, I/O, algorithms, external memory	**20030611 180**	15. NUMBER OF PAGES 5
		16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION ON THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

<u>MEMORANDUM OF TRANSMITTAL</u>

U.S. Army Research Office
ATTN: AMSRL-RO-BI (TR)
P.O. Box 12211
Research Triangle Park, NC 27709-2211

☐ Reprint (Orig + 2 copies)   ☐ Technical Report (Orig + 2 copies)

☐ Manuscript (1 copy)   ☒ Final Progress Report (Orig + 2 copies)

                                     ☐ Related Materials, Abstracts, Theses (1 copy)
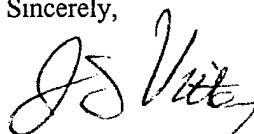
CONTRACT/GRANT NUMBER: *DAAD19-01-1-0725*

REPORT TITLE:

is forwarded for your information.

SUBMITTED FOR PUBLICATION TO (applicable only if report is manuscript):

Sincerely,

Final Progress Report

ARO Grant DAAD19–01–1–0725
External Memory Algorithms:
Dealing with MASSIVE Data

Prof. Jeffrey S. Vitter

Department of Computer Science
Duke University
Levine Science Research Center
Durham, NC 27708–0129
Email: jsv@cs.duke.edu

August 1, 2001 – February 28, 2003

1 Scientific Personnel

Faculty:

- Jeffrey S. Vitter, Professor

Research Associates

- Stergios Anastasiadis
- Rahul Shah

Graduate Students:

- Ankur Gupta

Visiting Scientists:

- Peter Varman (on sabbatical from Rice University)

2 Publications

[1] S. Anastasiadis, P. J. Varman, J. S. Vitter, and K. Yi. Lexicographically optimal smoothing for broadband traffic multiplexing. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, volume 21, Monterey, CA, July 2002.

[2] R. D. Barve and J. S. Vitter. A simple and efficient parallel disk mergesort. *ACM Trans. Comput. Syst.*, 35(2):189–215, March/April 2002.

[3] R. Grossi, A. Gupta, and J. S. Vitter. High-order entropy-compressed text indexes. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, January 2003.

[4] R. Grossi and J. S. Vitter. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. submitted to journal.

[5] J. S. Vitter. Algorithms and data structures for external memory. In *The Computer Engineering Handbook*, chapter 32, pages 32-1–32-33. CRC Press and IEEE Press, 2002.

[6] M. Wang, J. S. Vitter, L. Lim, and S. Padmanabhan. Wavelet-based cost estimation for spatial queries. In *Proceedings of the International Symposium on Spatial and Temporal Databases*, volume 7, Redondo Beach, CA, July 2001.

3 Scientific Progress and Accomplishments

3.1 Introduction and Overview

Problems involving massive amounts of data arise naturally in a variety of disciplines, such as spatial databases, geographic information systems, text repositories, string databases, constraint logic programming, object-oriented databases, statistics, virtual reality systems, and computer graphics. NASA's Earth Observing System project, the core part of the Earth Science Enterprise (formerly Mission to Planet Earth), produces petabytes (10^{15} bytes) of raster data per year! A major challenge is to develop mechanisms for processing the data efficiently, or else much of it will be useless.

The bottleneck in many applications that process massive amounts of data is the I/O communication between internal memory and external memory. The bottleneck is accentuated as processors get faster and parallel processors are used. Parallel disk arrays are often used to increase the I/O bandwidth. The goal of this proposal is to deepen our understanding of the limits of I/O systems and to construct external memory algorithms that are provably efficient. The three measures of performance are number of I/Os, disk storage space, and CPU time. Even when the data fit entirely in memory, communication can still be the bottleneck, and the related issues of caching become important.

Theoretical work involves development and analysis of provably efficient external memory algorithms and cache-efficient algorithms for a variety of important application areas. In [5], we give a broad survey of the state of the art in the design and analysis of external memory algorithms and data structures. We address several batched and on-line problems, involving text databases, prefetching and streaming data from parallel disks, and database selectivity estimation. Our experimental validation uses our TPIE programming environment.

3.2 Research Results

3.2.1 Parallel Disks

Technology trends indicate that developing techniques that effectively use multiple disks in parallel in order to speed up the performance of external sorting is of prime importance. In [2], we look at the *simple randomized merging (SRM)* mergesort algorithm that we earlier showed to be the first parallel disk sorting algorithm that requires a provably optimal number of passes and that is fast in practice. Knuth (in the new edition of *The Art of Computer Programming*, Vol. 3: *Sorting and Searching*) recently identified SRM (which he calls "randomized striping") as the method of choice for sorting with parallel disks. In [2], we present an efficient implementation of SRM, based upon novel data structures. We give a new implementation for SRM's *lookahead forecasting* technique for parallel prefetching and its *forecast and flush* technique for buffer management. Our techniques

amount to a significant improvement in the way SRM carries out the *parallel, independent* disk accesses necessary to efficiently read blocks of input runs during external merging.

We present the performance of SRM over a wide range of input sizes and compare its performance with that of *disk-striped mergesort (DSM)*, the commonly used technique to implement external mergesort on D parallel disks. DSM consists of using a standard mergesort algorithm in conjunction with striped I/O for parallel disk access. SRM merges together significantly more runs at a time compared with DSM, and thus it requires fewer merge passes. We demonstrate in practical scenarios that even though the streaming speeds for merging with DSM are a little higher than those for SRM (since DSM merges fewer runs at a time), sorting using SRM is significantly faster than with DSM, since SRM requires fewer passes.

The techniques in this paper can be generalized to meet the load-balancing requirements of other applications using parallel disks, including distribution sort, multiway partitioning of a file into several other files. and some potential multimedia streaming applications.

3.2.2 XML Databases

The extensible mark-up language (XML) is gaining widespread use as a format for data exchange and storage on the World Wide Web. Queries over XML data require accurate selectivity estimation of path expressions to optimize query execution plans. Selectivity estimation of XML path expression is usually done based on summary statistics about the structure of the underlying XML repository. All previous methods require an off-line scan of the XML repository to collect the statistics.

In [6], we propose XPathLearner, a method for estimating selectivity of the most commonly used types of path expressions without looking at the XML data. XPathLearner gathers and refines the statistics using query feedback in an on-line manner and is especially suited to queries in Internet scale applications since the underlying XML repositories are likely to be inaccessible or too large to be scanned entirely. Besides the on-line property, our method also has two other novel features: (a) XPathLearner is workload aware in collecting the statistics and thus can be dramatically more accurate than the more costly off-line method under tight memory constraints, and (b) XPathLearner automatically adjusts the statistics using query feedback when the underlying XML data change. We show empirically the estimation accuracy of our method using several real data sets.

3.2.3 Streaming Algorithms

In [1], we investigate the problem of smoothing multiplexed network traffic, when either a streaming server *transmits data* to multiple clients, or a server *accesses data* from multiple storage devices or other servers. We introduce efficient algorithms for lexicographically optimally smoothing the aggregate bandwidth requirements over a shared network link. In the data transmission problem, we consider the case in which the clients have different buffer capacities but no bandwidth constraints, or no buffer capacities but different bandwidth constraints. For the data access problem, we handle the general case of a shared buffer capacity and individual network bandwidth constraints. Previous approaches in the literature for the data access problem handled either the case of only a single stream or did not compute the lexicographically optimal schedule.

Lexicographically optimal smoothing (*lexopt* smoothing) has several advantages. By provably minimizing the variance of the required aggregate bandwidth, maximum resource requirements within the network become more predictable, and useful resource utilization increases. Fairness in sharing a network link by multiple users can be improved, and new requests from future clients

4

are more likely to be successfully admitted without the need for frequently rescheduling previously accepted traffic. Efficient resource management at the network edges can better meet quality of service requirements without restricting the scalability of the system.

3.2.4 Space-Efficient Indexes for Text Databases

The proliferation of online text, such as on the World Wide Web and in databases, motivates the need for space-efficient text indexing methods that support fast string searching. In this scenario, consider a text T that is made up of n symbols drawn from a fixed alphabet Σ and that is represented in $n \log |\Sigma|$ bits by encoding each symbol with $\log |\Sigma|$ bits. The goal is to support quick search queries of any string pattern P of m symbols, with T being fully scanned only once, namely, when the index is created.

Text indexing schemes published in the literature are greedy of space and require additional $\Omega(n \log n)$ bits in the worst case. For example, suffix trees and suffix arrays need $\Omega(n)$ memory words of $\Omega(\log n)$ bits in the standard unit cost RAM. These indexes are larger than the text itself by a factor of $\Omega(\log_{|\Sigma|} n)$, which is significant when Σ is of constant size, such as ASCII or UNICODE. On the other hand, they support fast searching either in $O(m \log |\Sigma|)$ time or in $O(m + \log n)$ time, plus an output-sensitive cost $O(occ)$ for listing the pattern occurrences.

In [4], we present a new text index that is based upon new compressed representations of suffix arrays and suffix trees. It achieves $O(m/\log_{|\Sigma|} n + \log_{|\Sigma|}^{\epsilon} n)$ search time in the worst case, for any constant $0 < \epsilon \leq 1$, with at most $(\epsilon^{-1} + O(1)) n \log |\Sigma|$ bits of storage; that is, the index size is comparable to the text size in the worst case. The above bounds improve *both* time and space of previous indexing schemes. Listing the pattern occurrences introduces a sublogarithmic slowdown factor in the output-sensitive cost, giving $O(occ \log_{|\Sigma|}^{\epsilon} n)$ time as a result. When the patterns are sufficiently long, namely, for $m = \Omega((\log^{2+\epsilon} n)(\log_{|\Sigma|} \log n))$, we can use auxiliary data structures in $O(n \log |\Sigma|)$ bits to obtain a total search bound of $O(m/\log_{|\Sigma|} n + occ)$ time, which is optimal.

3.2.5 Entropy-Compressed Text Indexes

In [3] we continue our work on space-efficient indexes and present a novel implementation of compressed suffix arrays exhibiting new tradeoffs between search time and space occupancy for a given text (or sequence) of n symbols over an alphabet Σ, where each symbol is encoded by $\lg |\Sigma|$ bits. We show that compressed suffix arrays use just $n H_h + O(n \lg \lg n / \lg_{|\Sigma|} n)$ bits, while retaining full text indexing functionalities, such as searching any pattern sequence of length m in $O(m \lg |\Sigma| + \text{polylog}(n))$ time. The term $H_h \leq \lg |\Sigma|$ denotes the hth-order empirical entropy of the text, which means that our index is nearly optimal in space apart from lower-order terms, achieving asymptotically the empirical entropy of the text (with a multiplicative constant 1). If the text is highly compressible so that $H_n = o(1)$ and the alphabet size is small, we obtain a text index with $o(m)$ search time that requires only $o(n)$ bits. We also report further results and tradeoffs on on high-order entropy-compressed text indexes.

4 Technology Transfer

We plan to pursue the practical applications of space-efficient search indexes. Implementation is ongoing.

Our efforts are also having an impact internationally. We have had discussions about the feasibility of adding parallel disk capabilities to the LEDA project at Max Planck in Saarbruecken, Germany.